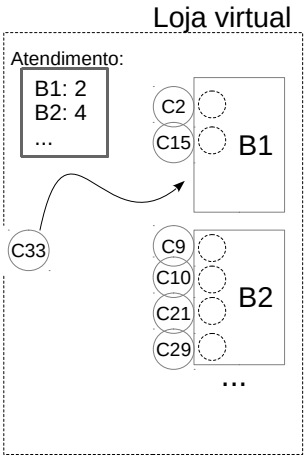
# Simulação de serviço de atendimento ao público

## Objetivos

O objetivo deste trabalho é possibilitar o treino de programação de aplicações Unix/Linux envolvendo processos e *threads*, respetivos meios de intercomunicação e primitivas de controlo de concorrência.
Para o efeito, propõe-se o desenvolvimento de uma aplicação que simule um serviço de atendimento público de clientes em balcões.

## Descrição geral




Numa loja virtual (ver figura junto), diversos balcões abrem para atendimento de clientes de um determinado serviço e operam durante um certo tempo.
Cada cliente que chega à loja dirige-se ao balcão com menor número de clientes em fase de atendimento e notifica o balcão da sua chegada; seguidamente é atendido, o que demora um tempo variável.
Um balcão permanece aberto durante um tempo que pode ser especificado. Quando o último balcão encerrar, a loja fecha.
O último balcão, antes de encerrar, recolhe os dados disponíveis e gera estatísticas tais como o número total de clientes atendidos na loja e em cada balcão, o tempo médio de atendimento por cliente, geral e por balcão e o tempo de abertura da loja e de cada balcão.

## Arquitetura da aplicação e restantes requisitos

A simulação do funcionamento da loja decorre mediante o arranque de dois tipos de programas, um ligado aos balcões de atendimento (**`balcao`**) e outro à geração de clientes (**`ger_cl`**). **`balcao`** é um programa *multithread* invocado na linha de comando tantas vezes quantas o número de balcões desejado; **`ger_cl`** é invocado da linha de comando uma vez e, por replicação, cria o número de processos cliente especificado na linha de comando.
Para a comunicação entre processos ou *threads* usam-se mecanismos como memória partilhada, *pipes* com nome (FIFOs), mutexes e variáveis de condição, conforme a situação em causa, e sempre de forma a evitar "esperas ativas" (*busy waiting*).

*Balcão:*
O programa tipo ligado aos balcões é invocado na linha comando, tantas vezes quantas o utilizador quiser, por:

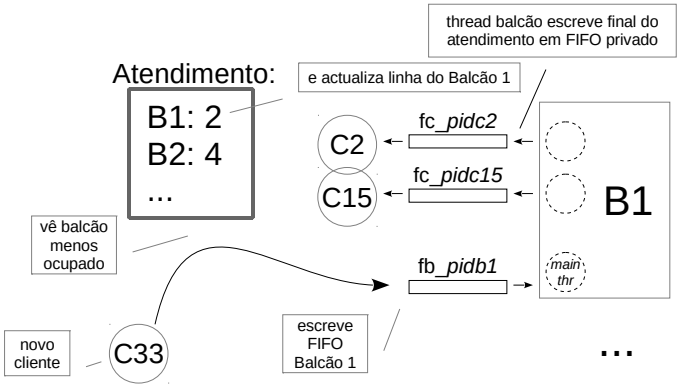
```
balcao <nome_mempartilhada> <tempo_abertura>
```

É um programa *multithread*, em que um novo *thread* é criado para atender um cliente e termina após esse atendimento.

O *thread* principal de uma instância de **`balcao`** inicia a sua operação verificando se a memória partilhada nomeada já existe e, se não existir, cria-a e inicializa as variáveis globais nela contidas. Note-se que esta fase de criação de memória partilhada exige a utilização de mecanismos de sincronização para se evitar situações de competição (*race conditions*) entre as diferentes instâncias de **`balcao`**. A estrutura da memória partilhada está descrita adiante.
Independentemente de quem criou e inicializou a memória partilhada, o *thread* principal de um **`balcao`**, terá de aceder à memória partilhada, com primitivas de sincronização adequadas, para disponibilizar a informação do balcão. Antes, cria um FIFO para atendimento de clientes. Na memória partilhada atualiza as variáveis relevantes, como a que guarda o nº de balcões disponíveis, e cria e preenche uma linha nova na Tabela de Balcões da memória partilhada (ver adiante a descrição desta tabela).
Seguidamente, o *thread* principal de balcao vai criar e iniciar as outras variáveis locais necessárias (e.g. variáveis de sincronização locais, necessárias ao acesso correto de todos os *threads* do processo à linha referente ao balcão, na Tabela de Balcões da memória partilhada global) e entrar em ciclo: coloca-se numa espera bloqueante no FIFO do balcão enquanto um novo cliente não chega e, aquando da chegada de

dados ao FIFO – que consistem na identificação do FIFO privado do cliente –, acorda e gera um *thread* de atendimento (passando-lhe como parâmetro a identificação do FIFO do cliente); depois, torna a bloquear no FIFO do balcão.
Cada programa **`balcao`** termina quando passar o **`tempo_abertura`** estipulado e completar os atendimentos em curso na altura; o FIFO de atendimento deve também ser removido. Se for o último balcão, gera as estatísticas referidas anteriormente e encerra a loja (e.g. remove a memória partilhada).

Cada *thread* de atendimento recebe do *thread* principal, como parâmetro, a identificação do FIFO privado do cliente, simula o atendimento adormecendo por um tempo em segundos igual ao nº de clientes do balcão mais uma unidade, mas sujeito a um máximo de 10 segundos, e escreve nesse FIFO "**`fim_atendimento`**", indicando que o atendimento foi concluído. Antes e depois de efetuar o atendimento, acede à linha desse balcão, na Tabela de Balcões, e atualiza a informação relevante, como o nº de clientes em atendimento ou o nº de clientes atendidos e o tempo médio de atendimento por cliente.




Nome do FIFO balcão: tem o formato "**`fb_`*`pid`***", em que ***`pid`*** é o identificador de sistema do processo **`balcao`**, e é relativo ao diretório **`/tmp`**.

*Cliente:*
O programa tipo associado aos clientes é invocado na linha comando uma só vez, por:

```
ger_cl <nome_mempartilhada> <num_clientes>
```

Se a loja ainda não estiver aberta (i.e., se a memória partilhada não existir), o programa termina; caso contrário, começa a gerar tantos clientes quantos os estipulados na linha de comando da seguinte maneira: cada cliente é um processo-filho (*clone*, gerado por **`fork()`**) que cria um FIFO privado e acede à Tabela de Balcões, partilhada, para ver os balcões abertos e número de cliente em atendimento em cada um (ver figura acima). Escolhe o balcão com menor número de clientes em atendimento e envia ao FIFO desse balcão a identificação do seu FIFO privado. Bloqueia neste FIFO, esperando a chegada da mensagem "**`fim_atendimento`**", e termina de seguida, eliminando antes o FIFO criado.

Nome do FIFO cliente: tem o formato "**`fc_`*`pid`***", em que ***`pid`*** é o identificador de sistema do processo cliente (diferente de **`ger_cl`**, por via do **`fork()`**...), e é relativo ao diretório **`/tmp`**.

*Memória partilhada:*
A memória partilhada, criada por uma instância de **`balcao`** e acedida de forma sincronizada por todos os processos balcão e cliente, é estruturada em duas partes (ver figura adiante):

- uma, contendo variáveis globais:
  - primitivas de protecção de acesso à zona;
  - o tempo de abertura da loja, em segundos desde a referência-tempo do sistema (*epoch*)
  - um inteiro que indica o nº de balcões registados;
- uma tabela, dita Tabela de Balcões, em que cada linha é criada e acedida pelos *threads* pertencentes ao **`balcao`** e em que as colunas são:
  - o número do balcão;
  - o tempo em que iniciou o funcionamento (em segundos desde a *epoch*)
  - a duração da abertura em segundos (nota: aqui, um valor de '**`-1`**' indica que o balcão ainda está aberto);
  - a identificação do FIFO em que o balcão atende clientes;
  - o nº de clientes em atendimento;
  - o nº de clientes atendidos;
  - o tempo médio de atendimento por cliente (em segundos).

*Memória Partilhada: exemplo369*

```
Data abertura da loja: 1543
N. total de balcões: 8
...primitivas sincronização...

Tabela de Balcões:

Balcao |    Abertura     |  Nome  |          Num_clientes          | Tempo_medio
 #     | Tempo | Duracao |  FIFO  | em_atendimento | ja_atendidos | atendimento
-------|-------|---------|--------|----------------|--------------|------------
 1     | 1543  | -1      | fb_123 | 2              | 123          | 23
 2     | 1883  | -1      | fb_554 | 4              | 17           | 39
 3     | 1745  | 820     | -      | 0              | 24           | 19
...
```

### *Ficheiro de Registos*

Todos os processos/*threads* deverão registar, de forma concorrente, num ficheiro de texto designado **`<nome_mempartilhada>.log`**, e situado no diretório de trabalho, linhas explicativas de todos os eventos relevantes para o decorrer da simulação. Isso possibilitará a observação do decorrer do programa e, mais importante, possibilitaria a confirmação das estatísticas finais, aquando do encerramento da loja.

Os registos devem ter o formato apresentado a seguir. A eventos diferentes dos exemplificados poderão ser dados novos nomes, devendo ter uma sintaxe semelhante.

```
quando              | quem   | balcao  | o_que                 | canal_criado/usado
-------------------------------------------------------------------------------------
2015-04-24 20:03:56  | Balcao | 1       | inicia_mempart        | -
2015-04-24 20:03:58  | Balcao | 1       | cria_linh_mempart     | fb_123
2015-04-24 20:04:16  | Balcao | 2       | cria_linh_mempart     | fb_554
2015-04-24 20:04:55  | Client | 1       | pede_atendimento      | fc_32456
2015-04-24 20:04:56  | Balcao | 1       | inicia_atend_cli      | fc_32456
2015-04-24 20:05:01  | Balcao | 3       | cria_linh_mempart     | fc_5632
2015-04-24 20:04:59  | Balcao | 1       | fim_atend_cli         | fc_32456
2015-04-24 20:04:55  | Client | 1       | fim_atendimento       | fc_32456
…
2015-04-24 20:09:50  | Balcao | 7       | fecha_loja            | fc_11889
2015-04-24 20:09:51  | Balcao | 7       | fecha_balcao          | fc_11889
```

## Entrega do trabalho

- Data limite para a entrega do trabalho: **2015/05/24, às 23:55h**.
- A forma de organização dos ficheiros deverá ser semelhante à indicada para o 1º trabalho prático, publicada, na página de "Sistemas Operativos", no Moodle da Universidade do Porto. Deve ser incluída uma *makefile* para a compilação dos programas.
- Deverá ser incluído um exemplo de um ficheiro de registos, com indicação das condições em que foi obtido (comandos usados na criação de balcões e clientes).